Forensic Examination of a RIM (BlackBerry) Wireless Device

June, 2002

Michael W. Burnette
Director of Information Technology
Rogers & Hardin LLP
mwb@rh-law.com

Table of Contents

Disclaimer 3
Introduction 3
Relevance of RIM forensics 3
The Hardware 4
Toolbox 4
Acquisition 4
BlackBerry Software Development Kit[9] 6
Evidence Collection 7
  Gathering Logs 7
    Unit Control Functions 7
  Imaging and Profiling 12
Evidence Review 17
  Hex Editor 17
  Simulator 18
Data Hiding 24
Conclusion 24
Acknowledgements 25
Glossary 26
Selected Bibliography 26

Disclaimer

This paper assumes that the investigator has secured appropriate authorization to intercept or access the files and information contained on or received by the electronic device in question, and that the implementation of any of the techniques and procedures set forth herein is being done under circumstances and restrictions that are in full compliance with The Electronic Communications Privacy Act of 1986 and other potentially applicable federal and state laws. No representation is made or intended that any specific application of the techniques and procedures set forth herein may be lawfully performed in any particular factual circumstance. Each investigator should secure appropriate legal advice with respect to each such application.

Introduction

This document is intended to familiarize the investigator with various methodologies and tools available to perform a forensic examination of a RIM (BlackBerry) device. The procedures and tools presented are by no means all encompassing, but are intended to elicit design of custom tools by those more programmatically inclined. The methods contained within have been tested using an Exchange Edition RIM pager model 950 and an Exchange Edition RIM handheld model 957. Desktop software will not be covered here. An investigator wishing to get a start should search the pumatech.com website for their white paper on reading Intellisync history and log files. It is most helpful.

There are two types of RIM devices within each model class. The Exchange Edition is meant for use in a corporate environment while the Internet Edition works with standard POP email accounts. The Exchange Edition employs Triple-DES encryption to send and receive but the Internet Edition communicates in clear text [1]. Neither employs an encrypted files system; hence the subject of this paper.

Relevance of RIM forensics

The RIM device shares the same evidentiary value as any other Personal Digital Assistant (PDA). As the investigator may suspect of most file systems, a delete is by no means a total removal of data on the device. However, the RIM's always-on, wireless push technology adds a unique dimension to forensic examination. Changing and updating data no longer requires a desktop synchronization. In fact, a RIM device does not need a cradle or desktop connection to be useful. The more time a PDA spends with its owner, the greater the chance is that it will more accurately reflect and tell a story about that person. Thus, the RIM's currently unsurpassed portability is the examiner's greatest ally.

The Hardware

The RIM device is designed around an Intel 32-bit i386 processor, a low power embedded version of the same processor that used to power a desktop PC. Each unit has 512 KB of SRAM and 4 or 5 MB of Flash RAM, depending on the model. The RIM's SRAM is analogous to the RAM on a desktop and the Flash memory is the "disk space" used to store the Operating System (OS), applications, and the file system. The RIM's OS is a single executable named PAGER.EXE and the applications are DLL's.

Toolbox

To test the methods presented in this paper, have the following tools handy:

- BlackBerry Desktop Software available free at www.blackberry.com
- BlackBerry C++ Software Development Kit v2.1 available free at www.blackberry.com
- Hex editor
- Text editor
- AA batteries
- Spare BlackBerry Cradles

The examination PC should meet the minimum requirements for the BlackBerry Software Development Kit (SDK) and have two available external 9-pin RS232 serial ports. Disk space required for evidence gathering is minimal: space equal to the amount of Flash RAM in the RIM units being investigated.

Acquisition

If the RIM is off, leave it off
If the RIM is on, turn the radio off
If the RIM is password protected, get the password

The BlackBerry is an always-on, push messaging device. Information can be pushed to the unit through its radio antenna at any time, potentially overwriting previously "deleted" data. Without warning, applications such as the email client, instant messaging, wireless calendar, and any number of third party applications may receive information which makes the forensic investigator's attempts to obtain an unaltered file system much more difficult. In order to preserve the unit, turn the radio off. Why not just turn the entire unit off? Three reasons: (1) The BlackBerry is not really "off" unless power is removed for an extended period of time or the unit is placed in data storage mode. Only the display, keyboard, and radio are shut down when using the GUI to turn off the unit. (2) When the unit is turned back on from an "off" or true powered down state, queued items may be pushed to the unit before there is a chance to turn the radio off. (3) A program might be installed on the unit that can accept remote commands via email (see http://www.onsettechnology.com). Under certain circumstances the original owner may prefer an investigator not view the information on a BlackBerry, and use this software to alter or delete information.

Radio ON

Radio OFF

If the unit is off at the time of acquisition, take it to a shielded location to turn it on and immediately shut down the radio before examination. A file cabinet or safe should block the signal well enough to provide time to work with the radio. Sometimes an interior room will do. It is ideal to have a unit on each network available to test local coverage (or lack thereof) for a safe location.

The Blackberry does not rely on an external power source so the examiner has to be diligent in making sure the unit does not power down as the result of a dead battery before the examination is complete. The 857/957 models are straightforward; just keep the unit in a cradle plugged into AC current. The 850/950 models require a AA battery which can be tricky to change. If the AA battery is changed fast enough, the charge stored in the unit will keep it running. Take too long and the unit will enter the radio/LCD/screen-off mode and eventually enter a full power-down mode. The radio/LCD/screen-off mode will cause changes to the system logs. The more destructive full power-down mode will cause a reset at startup wiping out most of the logs and initiating a system cleanup. A system cleanup has the forensically undesired effect of reclaiming unallocated (previously deleted) files by wiping and reorganizing sections of the file system.

**Be advised that serial communication is the fastest way to drain a battery so keep an eye on power levels throughout the examination.

Make note that completely powering off the RIM will wipe data from the SRAM. Logs stored there, which may be of interest, will not survive a full power-down. The catch-22 in investigating the RIM, using available tools, is that a file system snapshot ends in the same destructive reset that will wipe some of the logs, but investigating the logs could quite probably cause unwanted changes to the file system as well. While this author isn't aware of an application available to snapshot the SRAM along with the file system, it is important to practice good forensic method for when there is. Become familiar with the contents of the logs and their applicability to your case.

If the RIM is password protected, get the password. The methods in this paper use RIM provided software that will not circumvent password protection. The password itself is not stored on the unit; rather an SHA-1 hash of the password is stored and compared to a hash of what is entered. The examiner only has the opportunity to guess 10 times before a file system wipe occurs to protect the data. This wipe will destroy all non-OS files. No software exists to circumvent the password protection. A direct-to-hardware solution will be required if the password is not available.

BlackBerry Software Development Kit[9]

The BlackBerry Software Development Kit currently contains the best tools for examining a RIM. In that sense alone they can be considered forensic. Certainly an imaging tool that results in a reset leaves something to be desired! The kit is downloadable after filling out a web form at www.BlackBerry.net. Install it, view the contents of the subdirectories, and read through the help files before starting. The applications used to examine the unit are the Simulator (Simulator.exe and OSLoader.exe) and the Program Loader (programmer.exe).

The Program Loader will assist in gathering OS and application version numbers, memory locations, partition tables, SRAM maps, and the Holy Grail hex dumps (images). To read about all of the available switches for programmer.exe, refer to the Developers Guide.

The Simulator can be used to emulate the different types of RIM wireless units and the networks on which they function. This is great for scenario-testing, but for the forensic examiner provides a decent platform on which to examine a live copy of another RIM. In order to use the emulator to mimic a real unit, be able to identify the model and network type to simulate. The scope of this paper includes the 850/950, 857/957, and DATATAC/Mobitex networks, since that is what is supported by version 2.1 of the BlackBerry SDK. The networks differ in the structure and amount of data that is passed via the radio interface, thus each type requires a different set of DLLs to run. The DLLs are provided with the SDK. Once an image is acquired, there will be an opportunity to load it into the Simulator and view it as it would be on the original unit.

Evidence Collection

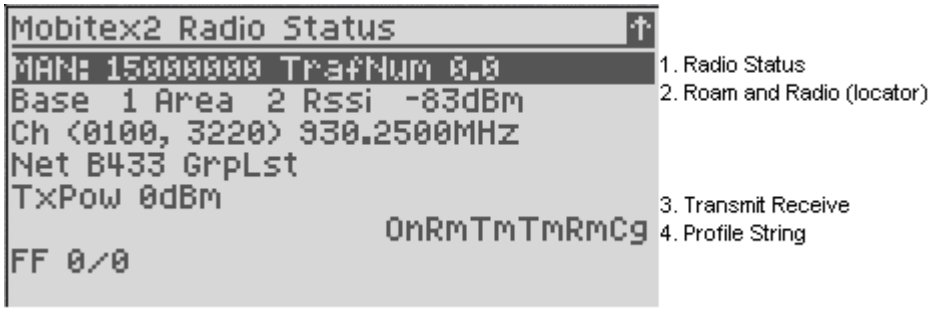
Gathering Logs

The first procedure of evidence collection violates the forensic method by requiring the investigator to record logs kept on the unit that will be wiped after an image is taken. The logs must be accessed on the original unit before the programmer SDK tool is applied. The logs are not accessed via the standard user interface. Rather, they are reviewed using the following hidden control functions.

Unit Control Functions

Radio Status

BlackBerry: Func + Cap + R
Simulator: Ctrl + Shift + R



The Mobitex2 Radio Status screen provides access to these four logs:

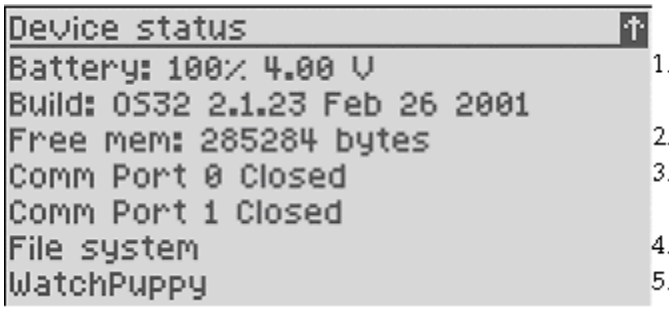
1. *Radio Status:* Enumerate the state of radio functions.
2. *Roam & Radio:* Records Base/Area (tower) and Roam (channel) information are recorded with duration of up to 99 hours per Base/Area/Channel. This log wraps at 16 entries and will not survive a reset. A blank entry represents a radio-off state.
3. *Transmit/Receive:* Records TxRx, gateway MAN addresses, type and size of the data transmitted, and both network and handheld date stamps per transmission.
4. *Profile String:* This is a recorded negotiation with the last utilized radio tower.

The Roam and Radio log is one of the most interesting. A call to a radio network provider or on-location-testing using a test RIM will provide the physical location that corresponds to each Base/Area. The investigator may be interested to know that the RIM in question has been to Houston, Texas in the last three days. Each entry is a record of

location at a single Base/Area location for up to 99 hours. The counter is always running, even when the radio is turned off, so to be sure to record these values as soon as possible to avoid log overwrites.

Device Status

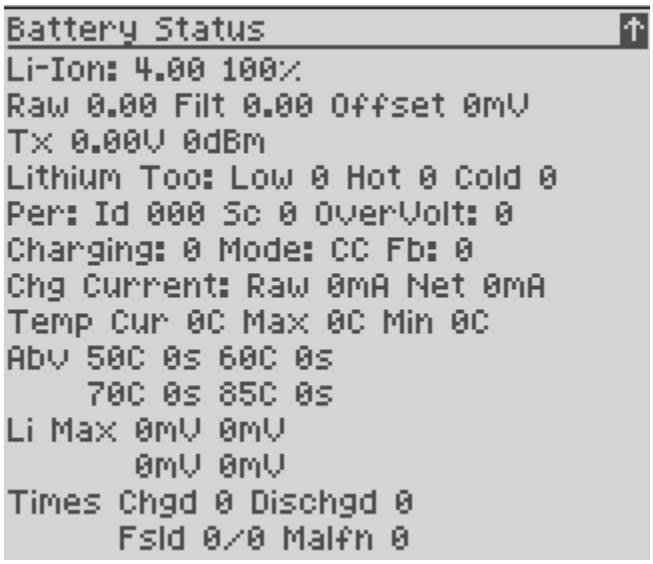
BlackBerry: Func + Cap + B (or V)
Simulator: Ctrl + Shift + B (or V)



1. Type, load, status, & Temp
2. Memory Allocation Info
3. Port status
4. File System allocation info
5. CPU WatchPuppy

Use the Rim's thumbwheel to choose a line in the Device status to see more information and to access logs.

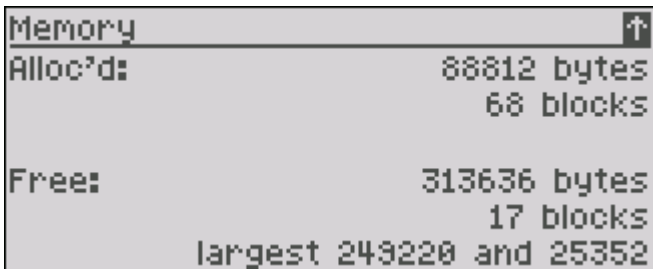
Battery Status


Battery Status
Li-Ion: 4.00 100%
Raw 0.00 Filt 0.00 Offset 0mV
Tx 0.00V 0dBm
Lithium Too: Low 0 Hot 0 Cold 0
Per: Id 000 Sc 0 OverVolt: 0
Charging: 0 Mode: CC Fb: 0
Chg Current: Raw 0mA Net 0mA
Temp Cur 0C Max 0C Min 0C
Abv 50C 0s 60C 0s
70C 0s 85C 0s
Li Max 0mV 0mV
0mV 0mV
Times Chgd 0 Dischgd 0
Fsld 0/0 Malfn 0

Battery Status provides information on battery type, load, status, and even temperature.

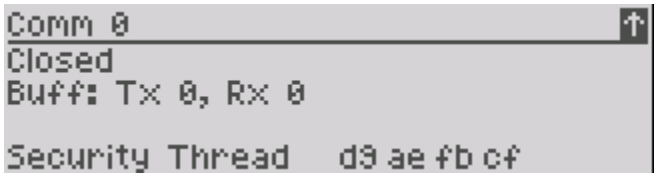
Free Mem

Memory



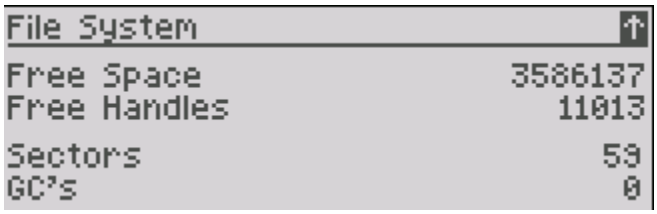
Free mem provides memory allocation information including the largest free blocks. Watch this value to prove that the unit is cleaning up the file system during a reset.

Comm Port



Comm port indicates port state. The security thread is not unique and its usefulness is not known.

File System



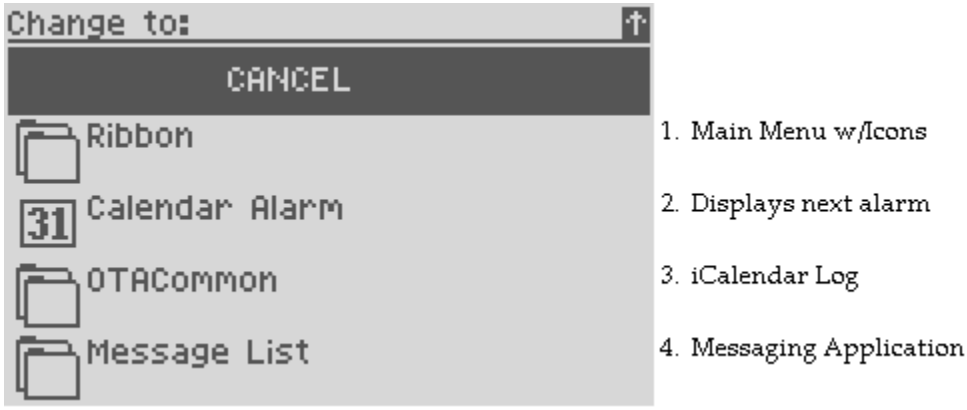
File System indicates basic values for free space and handles. There are limits on the number of handles, which can be found in the SDK guides.

WatchPuppy

The CPU WatchPuppy logs an entry when an application uses the CPU past a predetermined threshold. WatchPuppy kills processes that do not release the CPU.

Change to

BlackBerry: Func + Cap + S
Simulator: Ctrl + Shift + S



The only log of real interest in the Change To menu is the Over The Air (OTA) calendar log:

The OTA summarizes the last items synchronized via wireless calendaring on 32 lines and provides access to debugging information. The log can be emailed from the OTA Status menu but this is not advised since it will result in data being written to the file system.

Halt & Reset

BlackBerry: Func + Cap + Backspace
Simulator: Ctrl + Shift + Backspace

Pressing “R” or waiting a few seconds will cause a reset. This is analogous to using a paperclip to trip the reset button on the back of the unit! A reset will cause the unit to re-read the file-system and can trigger a file system cleanup. For now it should suffice to state that during a cleanup, items marked as deleted will be irrecoverably wiped and a basic “defragment” procedure can occur. It appears also that checksums are used to pinpoint bad Flash RAM areas. The point of a cleanup is to provide the largest blocks of free space as possible for future use. However, it is also exactly what the investigator hopes to avoid.

Imaging and Profiling

An image should be taken of the file system as the very first step as long the logs are not required OR a method of extracting the logs from the image is developed. An image, or bit-by-bit backup is acquired using an SDK utility that dumps the contents of the Flash RAM into a file easily examined with a hex editor. The Program Loader, which is used to perform most of the inspection in addition to taking the image, will cause a reset each time it is run. Recall a reset can mean a file system cleanup. This means to get a partition table, you risk changing the file system and spoiling the data. One way to work around this is to use the BATCH command described later. The BATCH command will group all command switches into one access so multiple resets can be avoided. The Program Loader is run from the command line:

PROGRAMMER [[-Pport] [-Sspeed] [-Wpassword **command**]]

Pipe the resulting screens to a text file using the standard DOS pipe at the end of the command line (>file.txt). The same holds true for the BATCH command.

SAVEFS

The SAVEFS command writes a hex dump of the RIM's Flash RAM to FILESYS.DMP, in the same directory as programmer.exe. The file will be exactly equal to the amount of Flash RAM available in the device (i.e. 950 = 4MB, 957 = 5MB). View this file with any hex editor. See appendix A for more hex dump information.

!! Immediately rename and write protect the file. The next time the Program Loader is run with SAVEFS it will overwrite FILESYS.DMP without warning. This is also a good opportunity to hash the file to prove integrity later in the investigation.

DIR

The DIR command lists applications residing on the handheld by memory location. This will be useful later when attempting to emulate the original handheld on a PC. Take note of any non-standard or missing applications.

```
Release -8
Bootrom: 1.0.13.0
Hardware: "RIM Inter@ctive Pager for Mobitex (Intel)"
        NAME         | FLASH  |  BASE   |  RAM    |  BASE   | THUNK  |  BASE   |
=====================+========+=========+=========+=========+========+=========+
PAGER950.EXE         |   30000| 3fc0000|        0|  580000|        0|        0|
---------------------+--------+---------+---------+---------+--------+---------+
Address.dll          |    fe00| 3fb0200|    1c08|  580000|      55c| 3ff6004|
AutoText.dll         |    4f00| 3fab300|     654|  581c08|      408| 3ff6560|
CryptoBlock.dll      |    a4e0| 3fa0e20|    162c|  58225c|      194| 3ff6968|
Database.dll         |    90e0| 3f97d40|      d0|  583888|       8c| 3ff6afc|
English.dll          |    8c50| 3f8f0f0|     1f6|  583958|       28| 3ff6b88|
Localization.dll     |    1610| 3f8dae0|       8|  583b4e|        0|        0|
Message.dll          |   2a510| 3f635d0|    3848|  583b56|      8fc| 3ff6bb0|
ribbon.dll           |   16a80| 3f4cb50|    1e5c|  58739e|      544| 3ff74ac|
SecureTransport.dll  |   19bb0| 3f32fa0|     d60|  5891fa|      5a8| 3ff79f0|
SerialDbAccess.dll   |    1890| 3f31710|     104|  589f5a|       d4| 3ff7f98|
Transport_MDP.dll    |    53c0| 3f2c350|    15a0|  58a05e|       fc| 3ff806c|
UI32.dll             |   1c810| 3f0fb40|     8ec|  58b5fe|       cc| 3ff8168|
Calculator.dll       |    3bf0| 3f0bf50|     4c0|  58beea|      190| 3ff8234|
Calendar.dll         |   29f00| 3ee2050|    263c|  58c3aa|      8cc| 3ff83c4|
MemoPad.dll          |    3280| 3ededd0|     118|  58e9e6|      300| 3ff8c90|
---------------------+--------+---------+---------+---------+--------+---------+
```

VER

The VER command lists applications residing on the handheld and corresponding version numbers. This will be useful later when attempting to emulate the original handheld on a PC. Take note of any non-standard or missing applications.

```
Hardware: "RIM Inter@ctive Pager for Mobitex (Intel)"

File Versions:

        NAME         |          VERSION          |    DATE    |   TIME   |
=====================+===========================+============+==========+
PAGER950.EXE         |     2,    1,   23,    0 |            |          |
---------------------+---------------------------+------------+----------+
Address.dll          |     2,    1,    2,   17 | 2001:08:23 | 17:26:22 |
AutoText.dll         |     2,    1,    2,   17 | 2001:08:23 | 17:26:03 |
CryptoBlock.dll      |     2,    1,    2,   17 | 2001:08:23 | 17:33:57 |
Database.dll         |     2,    1,    2,   17 | 2001:08:23 | 17:25:04 |
English.dll          |     2,    1,    2,   17 | 2001:08:23 | 17:36:16 |
Localization.dll     |     2,    1,    2,   17 | 2001:08:23 | 17:36:28 |
Message.dll          |     2,    1,    2,   17 | 2001:08:23 | 17:35:28 |
ribbon.dll           |     2,    1,    2,   17 | 2001:08:23 | 17:25:53 |
SecureTransport.dll  |     2,    1,    2,   17 | 2001:08:23 | 17:34:24 |
SerialDbAccess.dll   |     2,    1,    2,   17 | 2001:08:23 | 17:34:29 |
Transport_MDP.dll    |     2,    1,    2,   17 | 2001:08:23 | 17:33:46 |
UI32.dll             |     2,    1,    2,   17 | 2001:08:23 | 17:25:24 |
Calculator.dll       |     2,    1,    2,   17 | 2001:08:23 | 17:33:37 |
Calendar.dll         |     2,    1,    2,   17 | 2001:08:23 | 17:36:12 |
MemoPad.dll          |     2,    1,    2,   17 | 2001:08:23 | 17:36:36 |
---------------------+---------------------------+------------+----------+

API Versions:

Address.dll:
  Exports:   Address              2.1.2.17   -   Minimum Compatible: 2.0
  Imports:   Message              2.0
  Imports:   Options              2.0
  Imports:   UI32                 2.0
  Imports:   Ribbon               2.0
  Imports:   Database             2.0
  Imports:   AutoText             2.0

AutoText.dll:
  Exports:   AutoText             2.1.2.17   -   Minimum Compatible: 2.0
  Imports:   Database             2.0
  Imports:   UI32                 2.0
  Imports:   Options              2.0

CryptoBlock.dll:
  Exports:   CryptoBlock          2.1.2.17   -   Minimum Compatible: 2.0
  Imports:   Options              2.0
  Imports:   Database             2.0
```

MAP

The MAP command displays detailed Flash and SRAM maps.

```
Release -8
Hardware: "RIM Inter@ctive Pager for Mobitex (Intel)"

FLASH MAP    (available flash = 1152 KB)

  from   |   to    |    size | name
=========+=========+=========+====================
 3fc0000 | 3feffff |   30000 | PAGER950.EXE
---------+---------+---------+--------------------
 3fb0200 | 3fbffff |    fe00 | Address.dll
 3fab300 | 3fb01ff |    4f00 | AutoText.dll
 3fa0e20 | 3fab2ff |    a4e0 | CryptoBlock.dll
 3f97d40 | 3fa0e1f |    90e0 | Database.dll
 3f8f0f0 | 3f97d3f |    8c50 | English.dll
 3f8dae0 | 3f8f0ef |    1610 | Localization.dll
 3f635d0 | 3f8dadf |   2a510 | Message.dll
 3f4cb50 | 3f635cf |   16a80 | ribbon.dll
 3f32fa0 | 3f4cb4f |   19bb0 | SecureTransport.dll
 3f31710 | 3f32f9f |    1890 | SerialDbAccess.dll
 3f2c350 | 3f3170f |    53c0 | Transport_MDP.dll
 3f0fb40 | 3f2c34f |   1c810 | UI32.dll
 3f0bf50 | 3f0fb3f |    3bf0 | Calculator.dll
 3ee2050 | 3f0bf4f |   29f00 | Calendar.dll
 3ededd0 | 3ee204f |    3280 | MemoPad.dll
 3ed0000 | 3ededcf |    edd0 | [free]
---------+---------+---------+--------------------

RAM MAP    (available RAM = 560 KB)

  from   |   to    |    size | name
=========+=========+=========+====================
  580000 |  581c07 |    1c08 | Address.dll
  581c08 |  58225b |     654 | AutoText.dll
  58225c |  583887 |    162c | CryptoBlock.dll
  583888 |  583957 |      d0 | Database.dll
  583958 |  583b4d |     1f6 | English.dll
  583b4e |  583b55 |       8 | Localization.dll
  583b56 |  58739d |    3848 | Message.dll
  58739e |  5891f9 |    1e5c | ribbon.dll
  5891fa |  589f59 |     d60 | SecureTransport.dll
  589f5a |  58a05d |     104 | SerialDbAccess.dll
  58a05e |  58b5fd |    15a0 | Transport_MDP.dll
  58b5fe |  58bee9 |     8ec | UI32.dll
  58beea |  58c3a9 |     4c0 | Calculator.dll
  58c3aa |  58e9e5 |    263c | Calendar.dll
  58e9e6 |  58eafd |     118 | MemoPad.dll
  58eafe |  5f6fff |   68502 | [free]
  5f7000 |  60bfff |   15000 | PAGER950.EXE
---------+---------+---------+--------------------
```

ALLOC

The ALLOC command displays a "partition table" that lists the breakpoints between application memory and file system memory. Take note of any unused sectors and any difference between the end of the files area and the start of the OS and application area. These do not have to be the same and is an excellent example of how data hiding can occur on a RIM device. Review the -E, -D, and -A switches in the Developer's Guide for how to achieve this.

```
RIM Wireless Device Command-Line Programmer Version 1.0.0.28
Copyright 2000 Research In Motion Limited
Connecting to device
Enter password (10 left):
Connected

Flash Allocation Log

Last valid entry: 0
  Files:         45 sectors at 0x3c00000
  Unused:         0 sectors at 0x3ed0000
  OS and Apps:   18 sectors at 0x3ed0000
  Fixed use:      1 sectors at 0x3ff0000
  1K Blocks:     18
  4K Blocks:      2
```

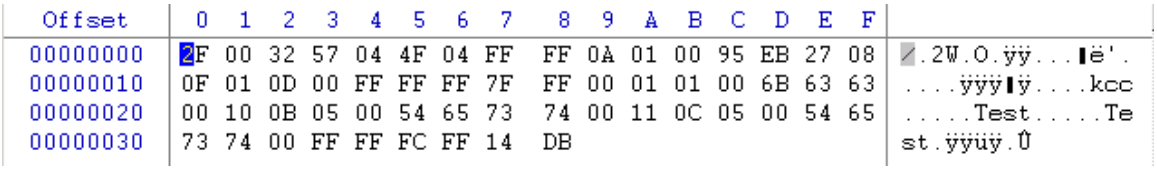
BATCH *filename*

The BATCH command groups the previous commands into a single communication session with the RIM device. This author's testing has shown that all of the commands are compatible within the same batch, with the exception of the SAVEFS or LOADFS options. These must be performed separately, which is why the SAVEFS image should come before all of the others. The amount of free space can possibly change during an initialization. Since a cleanup may erase previously retrievable data, it makes sense to perform the image first. Use the –Wpassword switch on the BATCH command line or on the first line of the batch file if a password is required.

Evidence Review

Two options are available for information review using the hex dump: (1) Manual review of the Hex file using a hex editor and (2) loading of the hex file into the BlackBerry SDK Simulator for review. The hex editor will provide access to the entire file system including deleted or "dirty" records indicated by byte 3 of the file header. Using the SDK will assist in "decoding" dates on extant records.

Hex Editor



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	2F	00	32	57	04	4F	04	FF	FF	0A	01	00	95	EB	27	08	/.2W.O.ÿÿ...ë'.
00000010	0F	01	0D	00	FF	FF	FF	7F	FF	00	01	01	00	6B	63	63	....ÿÿÿ.ÿ....kcc
00000020	00	10	0B	05	00	54	65	73	74	00	11	0C	05	00	54	65	.....Test.....Te
00000030	73	74	00	FF	FF	FC	FF	14	DB								st.ÿÿüÿ.Û

There is very little public information available regarding the bitwise file storage method used by the RIM OS. The figure above is taken from a file dump created using PROGRAMMER SAVEFS. It is a single email sent to user "kcc" with a subject of "Test" and a body of "Test." Dates are conspicuously missing, as at the time of writing this paper the format is still unknown. Likely the date is stored as standard C time (number of seconds since Jan 1, 1970). Certain structures are common to all "files" on a RIM. 10-bytes of overhead precede all records. This author has been able to determine the following of the 10-byte header:

Byte 1-2: Size of the record stored little endian. In the example above, the record size is 00 2F (hex) = 47 (dec). Count 47 + 10 bytes to capture the entire record plus header. The largest record (or sector size) is thus FFFFh or 65536 bits or 64K. This also happens to be the smallest addressable erasable block size.

Byte 3: This byte takes on only the following values: 12, 32, 52, 72, 92, and B2. It is used for what is known as "bit twiddling," which is the only case of a section of flash RAM being rewritten without first being erased. In flash memory, a "1" may be toggled to a "0" but not back again. In tests of the same file system, several hundred email messages were deleted, which resulted in a large decrease in 32 and a large increase in 12. A file can be deleted by toggling this value from 32 to 12, but moving it back again will not "undelete" it.

Byte 4-5: Record number stored little endian. All records are written with a unique identifier. Obviously the number of records would be limited to 65536 were it not for periodic file system cleanups. In the example above, the record number is 04 57 (hex). The next record written by the file system would be 0458, then 0459 and 045A… The

record number is contiguous unless fragmentation occurs within the file system. In this case, the fragmentation occurs in one direction only (forward) but will wrap around to the beginning of the address space once the end is reached.

Byte 6-7: Unknown, appears to be address or offset. May be map to next fragment of same record or a checksum.

Byte 8-10: Overhead footer, almost always FF FF 0A. This can be used to locate header.

The information contained within each record will vary according what application uses it for storage. As an example, an email record is easily readable between the binary. The "From", "To", and message body can be copied directly out of the hex file. Use the record header information to ensure you have an entire record. As you will read later, you may find partial email in "deleted" records. This is due to fragmentation and how the messaging application tends to "regroup" email into one large block later in the file system. When a partial email is found, as in more common file systems, the probability that the other parts as well as the concatenated copy will be found depends on the level of fragmentation and free space available. The "fragments" of email will be marked dirty and as long as the RIM owner has not deleted the email, the concatenated or "whole" version will be marked clean.

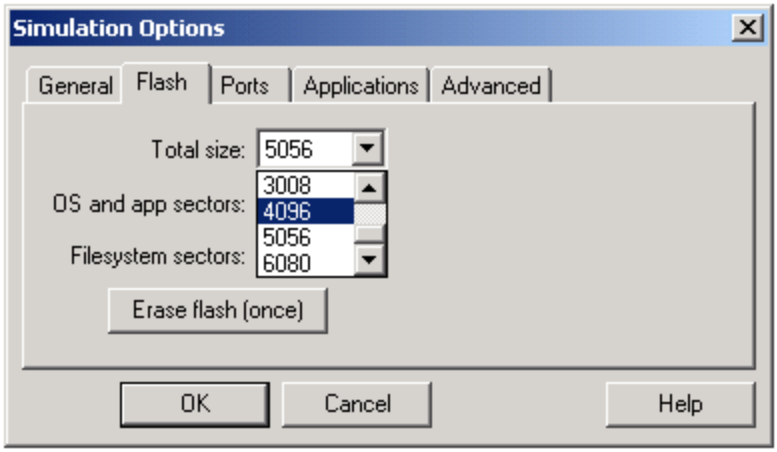
Simulator

To get the most from your hex dump without handling the original unit, load the dump file into the BlackBerry SDK Simulator. To accomplish this, rename the FILESYS.DMP file according to the following build rules:

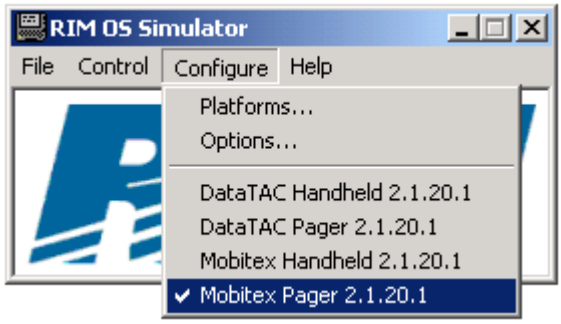
"FS"
"HH" if an 857/957 "Pgr" if an 850/950
"Mb" if Mobitex or "Dt" if Datatac
".DMP"

A Mobitex pager style BlackBerry would then have a load file of "FSPgrMb.DMP". If this file is placed in the same directory as the Simulator at the time of load, and all ancillary Simulator options are set to match, it will be substituted for the default blank file system. The file must not be marked read-only so use a copy. When exiting the simulator, the DMP file will be overwritten to match the last state of the simulator

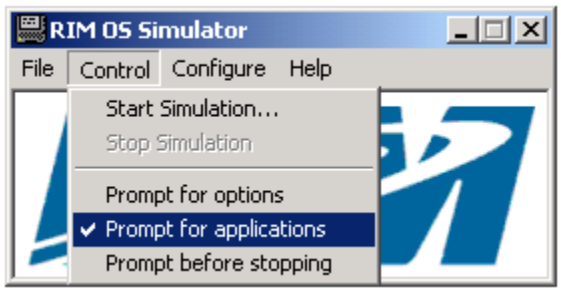
The Simulator must be set to match the Flash memory size to the size of the DMP file exactly. The file size cannot be larger than the available Flash. However, it is possible to use a file that is smaller; FFh will be appended to the image file to make it match the size set in the simulator.



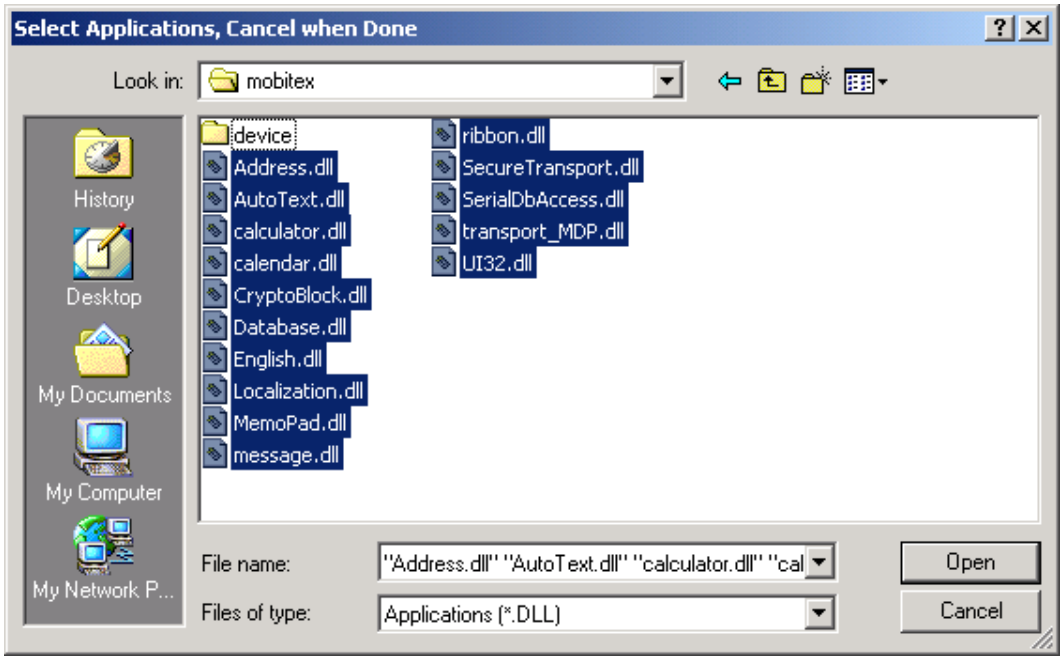
The Simulator then needs to be set to match the network and model tof the investigated unit.



Applications must be loaded from those available in the SDK. For this reason the DIR listing acquired in the earlier evidence acquisition will come in handy.



In this example, a full complement of default applications are loaded for a Mobitex BlackBerry. Some applications are written specifically for one model or the other; however, the default apps are not.



Run the simulator by choosing “control,” “start simulation.” If any prompt settings are checked on the control file menu, the system asks for the options above to be set one more time. The Simulator operates in exactly the same manner as a handheld BlackBerry with the additional convenience of PC keyboard manipulation.

It is also possible to connect the Simulator to a serial port on a PC. Run the following from a command line:

OSLoader.exe OsPgrMb.dll /s1

The DLL is for the 950. Substitute OsHhMb.dll for the 957. This command line specifies that the simulator will load the interface for the 950 and connect to serial port 1. A window will pop up asking which applications to load, select all of the DLL files under the \dll\mobitex or \dll\datatac directories and click "open." When the window opens again, press cancel.

Now attach the BlackBerry desktop software to the simulator through a NULL modem cable. Use a NULL modem cable to connect the two serial ports on the back of the PC. BlackBerry SDK support suggests model number SCNM9FF made by StarTech (www.startech.com). With the Simulator running from the command line on serial port 1, start the desktop software and configure it to use serial port 2. The desktop software will detect the "handheld" and prompt you to switch. It will take a while and look as if nothing is happening, but eventually control of the desktop will be regained and then Intellisync as well as the backup & restore can be used. It may be necessary to try a few PCs before finding one which this actually works on. This author had three different issues on three different PCs with the serial ports alone. In one case the cradle being used had to be removed and replaced after each session to function properly. As a general rule, test first, and then investigate.

The File System[3][4][11]

The RIM file system is abstracted to appear as a database to most available API's in order to simplify programming. This abstraction qualifies as a File Translation Layer (FTL) hiding what is really a quite complicated system of file management.[3] Under the hood of your RIM device is a standard Flash RAM used to store all non-volatile data.

Flash is organized similarly to DRAM and has the 65ns read performance to match. However, write performance is slower by a factor of nearly 1000x. Writing flash is a binary AND or NAND meaning each '1' in memory can be toggled to '0' but not back again without an erasure. However, an erasure can only occur in blocks of 64K while writing can occur in any interval. An erasure costs more in terms of time than a write due to the act of conditioning. Conditioning means forcing every '1' to a '0' before resetting (erasing) all bits to '1' again in order to prolong the life of the Flash RAM. [6]

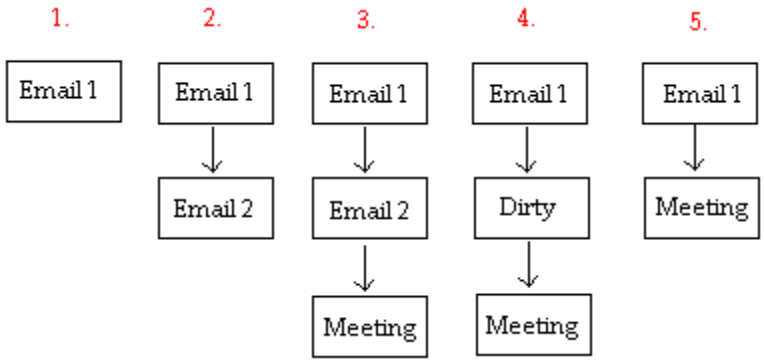
Reading: Occurs in any length and is very fast.

Writing: Comparatively slow one-way function in any length inhibited only by the amount of file system fragmentation.

Erase: Condition, Rewrite to 1's only in 65K blocks.

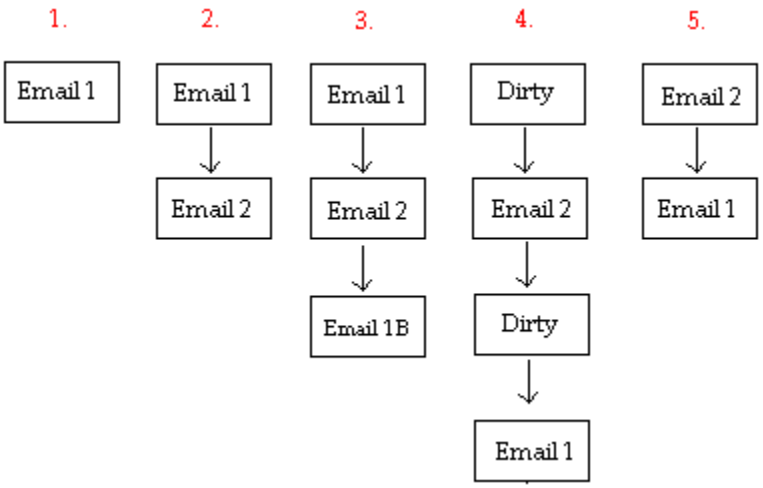
Rewrite: Save 64K block to SRAM, erase 64K block of Flash, change image in SRAM, write back to erased Flash block. This takes approximately 5 seconds to complete.

It goes without saying that the hardware is optimized for reads. The RIM OS makes the best use of the hardware available by implementing a log-based file system.[11] Files are written in a linked list one at a time, each being appended to the last or the "end of the log". Each file or record has its own unique identifier, a number between 0 and 65536. When a change is necessary to an existing record, the original record is marked as dirty (bit twiddling most likely)[4] and the new version is written to the end of the file system with a new unique identifier. This process eliminates the need for on-the-fly erasures which cost a great deal of time. Periodically, the OS will clean old records marked as dirty, and defragment the file system, if necessary, to allow for more room for the file system to grow (expand the log). Once the end of address space is reached, the log wraps back around to the beginning of the address space. Unlike traditional file systems, fragmentation occurs in one direction only. Even if the first part of a file is near the end of address space and the next part wraps back around to the beginning, the virtual address space is the log, which is in one direction only.[11]



1. Email 1 is received and written to the file system
2. Email 2 is received and appended to the file system
3. Item is added to the calendar and is appended to the file system
4. Email 2 is deleted
5. File system cleanup occurs at next reset or when out of space

The log based file system and its interaction with the standard applications has notable ramifications when it comes to recovering whole files that either cross 64K sector boundaries or for which storage has been written several times. Take for instance the case of receiving a large email:



1. The enterprise or SRP servers send the first block (size dictated by network) and wait for requests for more. This first block is written to the file system.
2. A second email is received and written to the file system.
3. The user requests "more" of the first email through the messaging GUI. The second block of the large email is written to the file system.
4. The first and second block are written together as a single record at the end of the file system and the original records are marked dirty.
5. File system cleanup occurs at next reset or when out of space. [9]

Data Hiding

Data hiding is accomplished in several different ways on a RIM device. Hidden databases, partition gaps, and obfuscated data are but a few. Presented here are merely a few ideas to get the investigator thinking in the right direction.

Custom written databases with no icon in the Ribbon graphical user interface (GUI) are capable of providing hidden data transport. A hacker may write a program that utilizes a database accessible only through device synchronization. The average user or uninformed investigator will never have knowledge of the "hidden" database. An application such as the Rim Walker[8] database reader can thwart such an attempt as it will provide access to all databases on a unit. Unfortunately, it will need to be installed on the unit investigated to function. Assuming clear text, unencrypted data storage the SAVEFS Programmer command produces a file system dump capable of review for such a database.

Information can also be entered directly into unused space in the file system using the SDK tools. As long as the information is placed sufficiently near the "end" of the available file system space, it will stay intact even through a device reset. This method can be tested using the SAVEFS Programmer command. Type a paragraph at the "end" of a SAVEFS dump file and then use the LOADFS Programmer command to reload it into a device. The data cannot be accessed via the interface, but can be viewed again using the SAVEFS Programmer command.

Data can be hidden in the gap between the OS/Application and Files partitions. Use The ALLOC Programmer command to view the partition table. By default, the two partitions converge at the same sector number leaving no gap. However, the size of these two partitions can be independently altered using ALLOC as long as any space removed from a partition is blank space. If a gap exists, it will appear as "unused" sectors in an ALLOC partition table. A hacker may write a program that uses direct memory access to write to the "gap" or simply use the SAVEFS and LOADFS commands to load data to those unused sectors.

Conclusion

This paper has presented topics on the study of the RIM device with a specific slant toward forensic investigation by trained personnel. All methods should be tested in a lab environment before being attempted in a live investigation. As is true in any forensic investigation, variations in PC Operating Systems, SDK Tools, RIM Operating Systems, wireless network types, and RIM model could have unforeseen consequences if not tested in the investigator's particular environment.

Acknowledgements

RIM and Research In Motion are registered trademarks of Research In Motion Limited. RIM, BlackBerry, and Research In Motion are trademarks of Research In Motion Limited. All other brands, product names and company names mentioned herein may be trademarks or registered trademarks of their respective owners.

Glossary

Little Endian - A computer architecture in which, within a given 16- or 32-bit word, bytes at lower addresses have lower significance (the word is stored "little-end-first"). Intel microprocessors and a lot of communications and networking hardware are little-endian. The term is sometimes used to describe the ordering of units other than bytes; most often, bits within a byte.

MPAK Time – Time stamped on packet traffic entering the network bound for the BlackBerry. Should be close but not exactly the time on the unit.

Over The Air Calendaring (OTA) – Refers to the calendar synchronization function available in later versions of the Enterprise Edition BlackBerry.

Watchpuppy – Operating system function that monitors running applications for overuse of the CPU. The BlackBerry is a cooperative multitasking environment, this service will shut down any application that uses too much of the CPU and does not yield enough control to the OS.

Selected Bibliography

This paper compiles information and/or concepts presented in the following works:

1. BlackBerry Crypto
http://csrc.nist.gov/cryptval/140-1/140sp/140sp137.pdf

2. RIM serial protocol
http://www.off.net/cassis/protocol-description.html

3. Implementing a Resident Flash Disk with an Intel386 EX Embedded Processor
http://developer.intel.com/design/intarch/applnots/292157.htm

4. Software Concerns of Implementing a Resident Flash Disk
http://www.intel.com/design/flcomp/applnots/292173.htm

5. Advantages of Large Erase Blocks
http://www.intel.com/design/flcomp/applnots/297637.htm

6. Transactions in a Flash
http://www-2.cs.cmu.edu/~mprice/masters96_flash.pdf

7. Dongleberry (Wiring)
http://rallypilot.sourceforge.net/bb/index.html

8. RimWalker

http://tucows.com

9. BlackBerry Software Development Documentation
http://www.blackberry.net/developers/na/c_plus/knowledge/search.shtml

10. BlackBerry SDK Developers guide v 2.1
http://www.blackberry.net/developers/na/c_plus/knowledge/documentation/download/usersguide.pdf

11. BlackBerry Operating System API v2.0
http://www.blackberry.net/developers/na/c_plus/knowledge/documentation/download/os_20_mobitex.pdf

12. BlackBerry Programming RIM Palm Sized Wireless Handhelds 2.0
http://www.blackberry.net/developers/na/c_plus/knowledge/documentation/download/handheld.pdf

13. BlackBerry Database API 2.0
http://www.blackberry.net/developers/na/c_plus/knowledge/documentation/download/database_mobitex.pdf